

Mifare RFID(RC522) 讀寫模組

一、產品概述

◆ 【RFID 模組簡介】

RFID-RC522 模組採用 NXP MF RC522 晶片設計讀卡電路，使用方便，適用於設備開發、讀卡器開發等應用的使用者及需要進行射頻卡終端設計/生產的使用者。模組採用電壓為 3.3V, 通過 SPI 介面，就可以直接與任何具備 SPI 介面的 CPU 主機板相連接與通信



◆ 【RC522 晶片簡介】

MF RC522 是應用於 13.56MHz 非接觸式通信中高集成度的讀寫卡晶片，是 NXP 公司推出的一款低電壓、體積小的非接觸式讀寫卡晶片，是智慧型儀表和可攜式手持設備研發的理想選擇。MF RC522 利用了先進的調變和解調概念，集成了在 13.56MHz 下所有類型的被動非接觸式通信方式和協定。支援 14443A 相容應答器信號。數位部分處理 ISO14443A Frame(幀)和錯誤檢測。此外，還支援快速 CRYPTO1 加密演算法。MFRC522 支援 MIFARE 系列更高速的非接觸式通信，雙向資料傳輸速率高達 424kbit/s。作為 13.56MHz 高集成度讀寫卡系列晶片家族的成員，MF RC522 與 MF RC500 和 MF RC530 有不少相似之處，同時也具備許多特點和差異。它與主機間通信採用 SPI 模式，有利於減少連線，縮小 PCB 板體積，降低成本。

【電氣參數簡介】

工作電流：13—26mA/直流 3.3V

空閒電流：10-13mA/直流 3.3V

休眠電流：<80uA

峰值電流：<30mA

工作頻率：13.56MHz

支援的卡類型：mifare1 S50、mifare1 S70、mifare UltraLight、mifare Pro、mifare Desfire

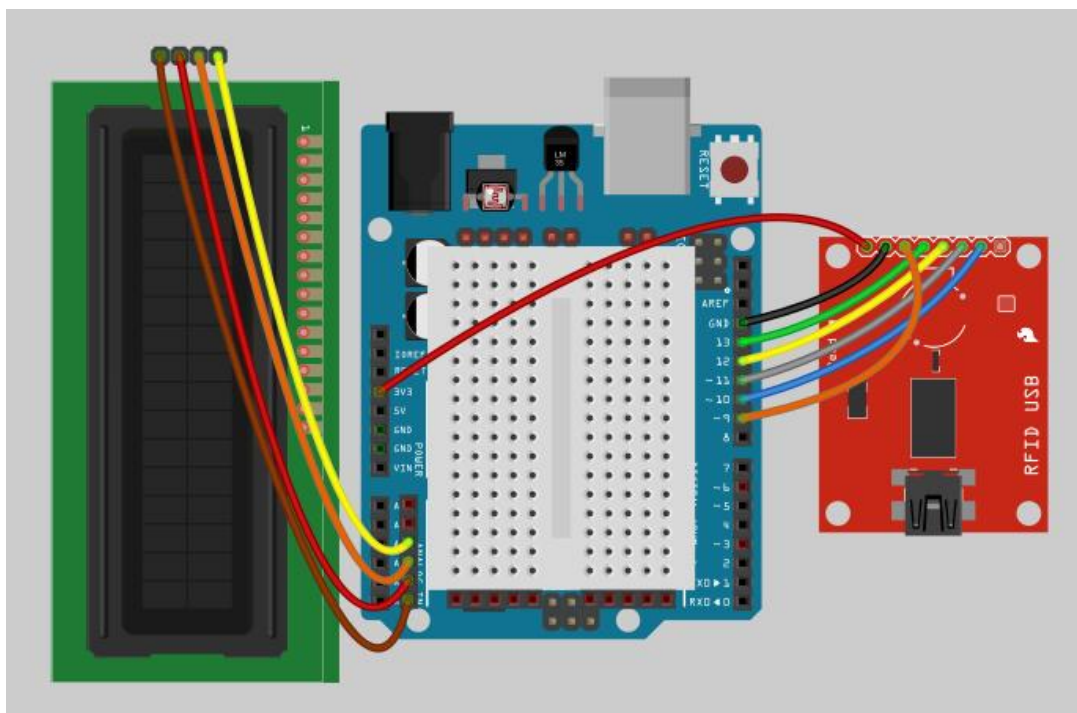
產品物理特性：尺寸：40mm×60mm

【模組介面 SPI 參數】

資料傳輸速率：最大 10Mbit/s

二、接腳說明(採用 SPI 介面)

接腳	名稱	RFID Reader(SPI 介面)
1	IRQ	X
2	NSS	接 Arduino D10
3	MOSI	接 Arduino D11
4	MISO	接 Arduino D12
5	SCK	接 Arduino D13
6	RST	接 Arduino D9
7	GND	接 GND
8	3.3V	接 3.3V



RFID 對應接腳接線圖

三、軟體常用函數使用說明(RFID Library)

- **RFID 類別的全域變數(global variable)**
unsigned char **serNum[5]** : 紀錄 RFID Tag 的 ID/序號(4 Bytes + 1 byte(檢查碼))
serNum [0..3]: serial number of the card
serNum [4]: XOR checksum of the serNum [0..3]
- **RFID(int chipSelectPin, int NRSTPD)** : RFID class constructor
ex : RFID rfid(10, 9); //宣告一個 RFID 類別的物件，名稱為 rfid 且其 NSS(CS) 與 RST Pin 分別接至 Arduino 的 D10 與 D9 腳位
- **bool isCard()** : 檢查是否偵測到 RFID Tag ? 並回傳 true/false
ex : if (rfid.isCard()) {
 } //若偵測到 RFID Tag 則處理下一步
- **void init()** : 初始化 RFID
ex : rfid.init(); //將 RFID 物件 rfid 初始化
- **bool readCardSerial()** : 讀取 RFID 的 ID/序號(4 bytes)與檢查碼(1 byte)
ex : if (rfid.readCardSerial()) {.... } ; //可用 rfid.serNum[0].. rfid.serNum[4] 取出序號
- **unsigned char anticoll(unsigned char *serNum)** : 處理當有多張卡進入讀寫器操作範圍時，防衝突機制會從其中選擇一張進行操作，並回傳處理狀態(status)與將此張卡的序號儲存於 serNum[0]~ serNum[3] Array 內
回傳處理狀態(status)有 :
 - MI_OK : 成功
 - MI_NOTAGERR
 - MI_ERRex : status = rfid.anticoll(rfid.serNum);
if (status == MI_OK) {.... } ; //可用 rfid.serNum[0].. rfid.serNum[4] 取出序號
- **unsigned char SelectTag(unsigned char *serNum)** : 選擇序號為 serNum[0..3]的卡片並回傳該卡片的記憶體容量(單位 : Bits)，當要處理卡片記憶體區塊的任何動作(讀/寫/加/減,...)前，皆須先透過此程序選擇一張卡片
ex: RC_size = rfid.SelectTag(rfid.serNum); // 選擇一張卡並回傳卡片的容量
if (RC_size != 0) {.... } //可印出卡片的容量(單位 : Bits)

- `unsigned char auth(unsigned char authMode, unsigned char BlockAddr, unsigned char *Sectorkey, unsigned char *serNum)`：處理卡片記憶體區塊的任何動作(讀/寫/加/減,...)前，皆須先透過密碼的驗證程序，通過後才能操作記憶體區塊資料

輸入參數如下：

`authMode`：驗證密碼模式，PICC_AUTHENT1A 指使用密碼 A(Key A)驗證，PICC_AUTHENT1B 指使用密碼 B(Key B)

`BlockAddr`：區塊編號(0 ~ 63)

`Sectorkey`：密碼儲存位址

`serNum`：卡片序號儲存位址

回傳處理狀態(status)：

- `MI_OK`：成功
- `MI_NOTAGERR`
- `MI_ERR`

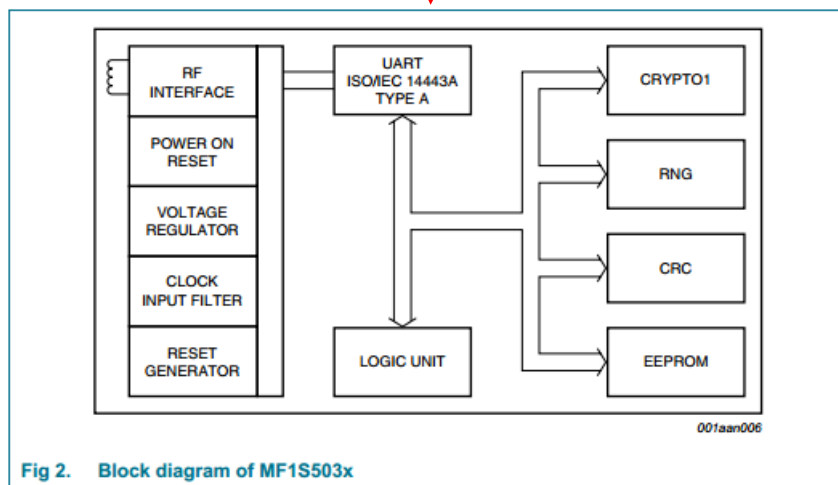
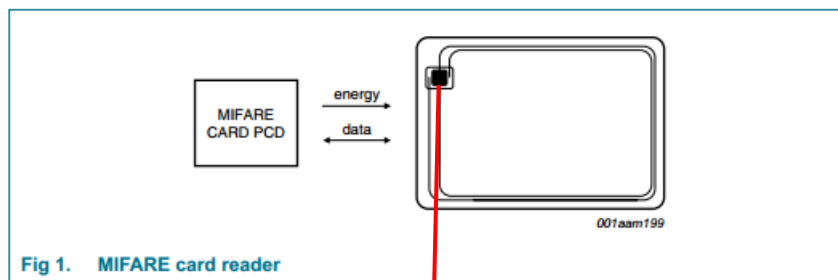
```
ex : status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],  
                        rfid.serNum); // Authentication by verify Key A  
    if (status == MI_OK) { Read or write block ,... }
```

- `unsigned char read(unsigned char blockAddr, unsigned char *recvData)`：讀取 RFID 所指定 `blockAddr`(0 ~ 63)的整個區塊資料(16 bytes)並將其儲存於 `recvData` 的字元串中
ex : `rfid.read(11,recvData)` ; //讀取 RFID Tag block 11 的資料(16 bytes)

- `unsigned char write(unsigned char blockAddr, unsigned char *writeData)`：將 `writeData` 字元字串(16 Bytes)中的資料寫入 RFID 所指定的 `blockAddr`
ex : `rfid.write(11,writeData)` ; //將 `writeData` 的資料(16 bytes)寫入 Tag 的 block 11

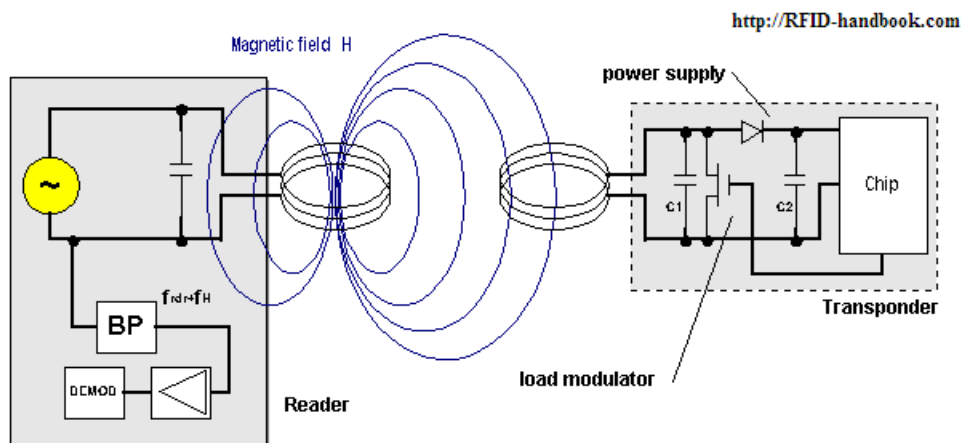
- `void halt()`：讓卡片進入休眠狀態
ex : `rfid.halt()`; //設定卡片進入休眠狀態

四、Mifare Classic Tag(MF1S50)/1K bytes 架構



◆ 運作原理

讀寫器(Reader)向 RFID Tag(M1 卡)發一組固定頻率的電磁波,卡片內有一個 LC 串聯諧振電路,其頻率與讀寫器發射的頻率相同,在電磁波的激勵下,LC 諧振電路產生共振,從而使電容內有了電荷,在這個電容的另一端,接有一個單向導通的二極體(Diode),將電容內的電荷 送到另一個電容內儲存,當所積累的電荷達到一定電壓 ($\sim 2V$)時,此電容可做為 MF1S50 IC 的控制單元的電源,並根據相應區的有效存取位元(access bits)來控制將卡內資料發射出去或讀取讀寫器的資料。其運作程序如下所述 :



- 請求應答(Answer to request)

RFID Tag 的通訊協定和通訊串列傳輸速率是定義好的,當有卡片進入讀寫器的操作範圍時,讀寫器會以特定的協定與它通訊,從而確定該卡是否為相容的卡,即驗證卡片的類型。而卡片也會依請求代碼發送回應 ATQA 訊號

- 防衝突機制 (Anticollision Loop)

當有多張卡進入讀寫器操作範圍時,防衝突機制會從其中選擇一張進行操作,未選中的則處於準備模式等待下一次選卡,該過程會返回被選卡的序號。

- 選擇卡片(Select Tag)

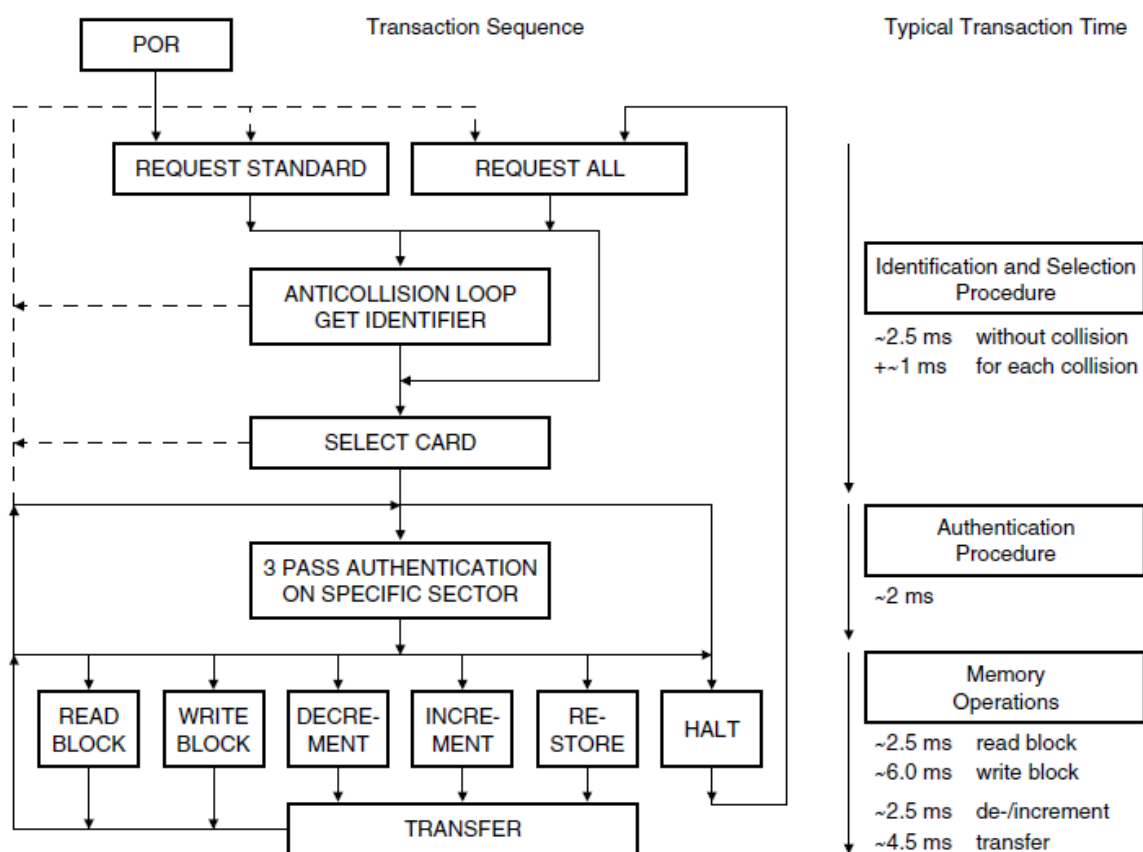
讀寫器使用選擇卡命令,選擇其中一張卡的序號進行確認和記憶體相關操作,此時卡回傳 ATS 碼(Answer To Select =08h),讀寫器透過 ATS 可以確定被選中的卡的類型,並同時返回卡的容量代碼。

- 三次互相授權認證機制(3 Pass Authentication)

選定要處理的卡片之後,讀寫器就確定要存取的區塊號碼(Block #),並對該區塊密碼進行密碼校驗,在三次相互認證之後就可以通過加密進行通訊。

- 區塊(Block)的操作方式(在執行任何區塊操作前，卡必須先要被選擇並經過碼驗證機制)

操作方式	描述	區塊類型(Block type)
讀 (Read)	讀一個區塊(Block)	資料、數值與 Trailer 區塊
寫 (Write)	寫一個區塊(Block)	資料、數值與 Trailer 區塊
加 (Increment)	對數值區塊(value block)進行加值	數值區塊(value block)
減 (Decrement)	對數值區塊進行減值	數值區塊(value block)
存儲 (Restore)	將區塊中的內容存到資料暫存器中	數值區塊(value block)
傳輸 (Transfer)	將資料暫存器中的內容寫入區塊中	數值區塊(value block)

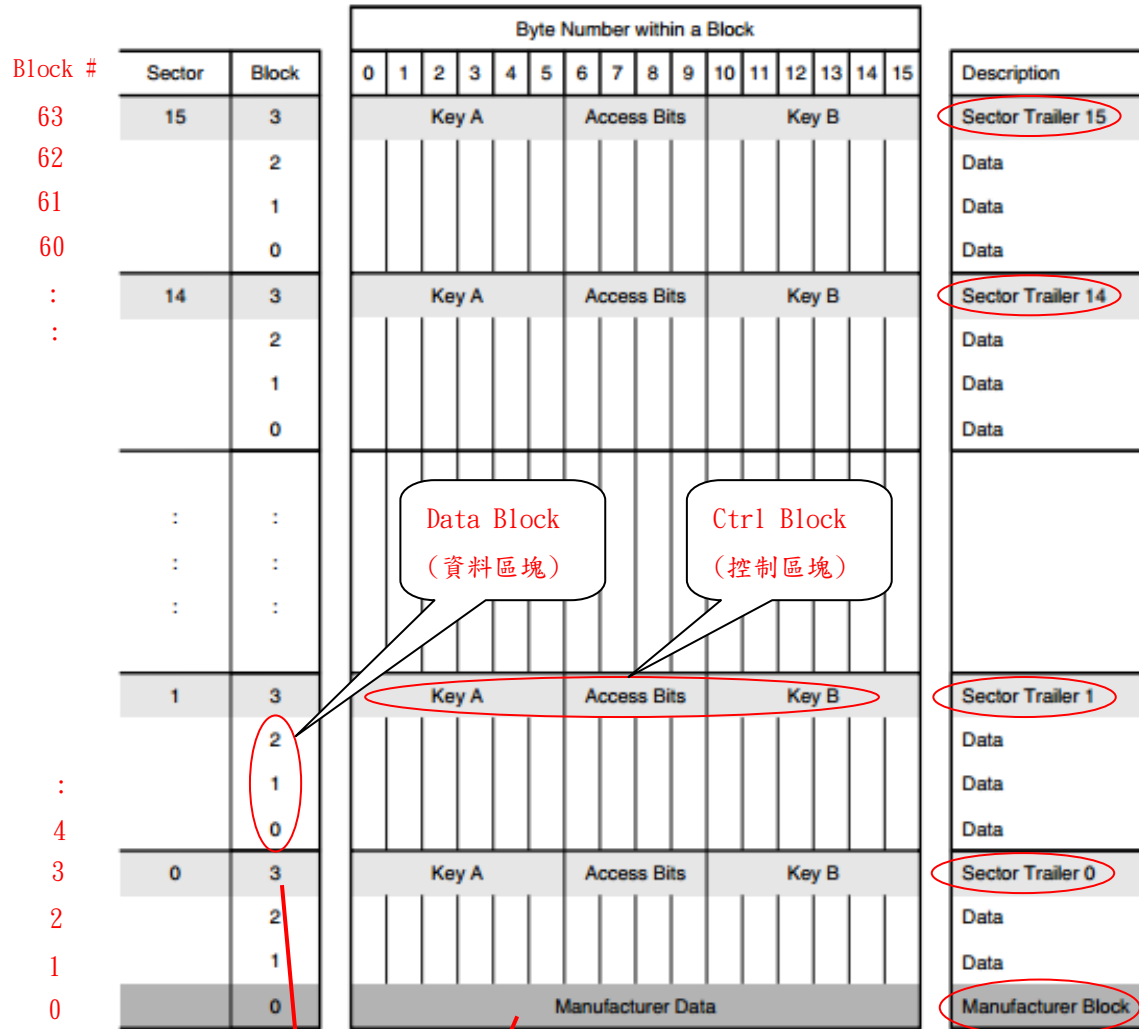


RFID Tag 的運作程序圖

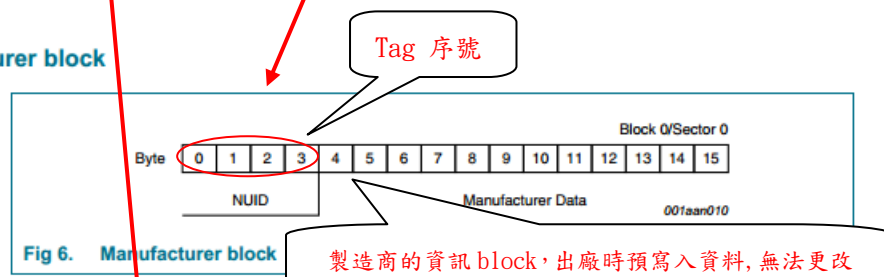
五、Mifare Classic Tag(MF1S50)/1K bytes 記憶體資料格式(16 Sec x 4 Blk x 16 bytes)

Memory organization

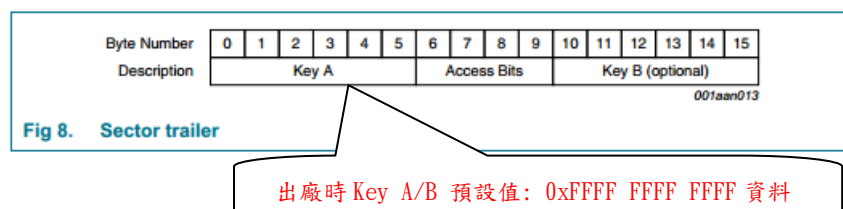
The 1024 × 8 bit EEPROM memory is organized in 16 sectors of 4 blocks. One block contains 16 bytes.



Manufacturer block



Sector trailer (block 3)



◆ 記憶體區塊簡介

– 記憶體區塊(Block) 有三種型態：

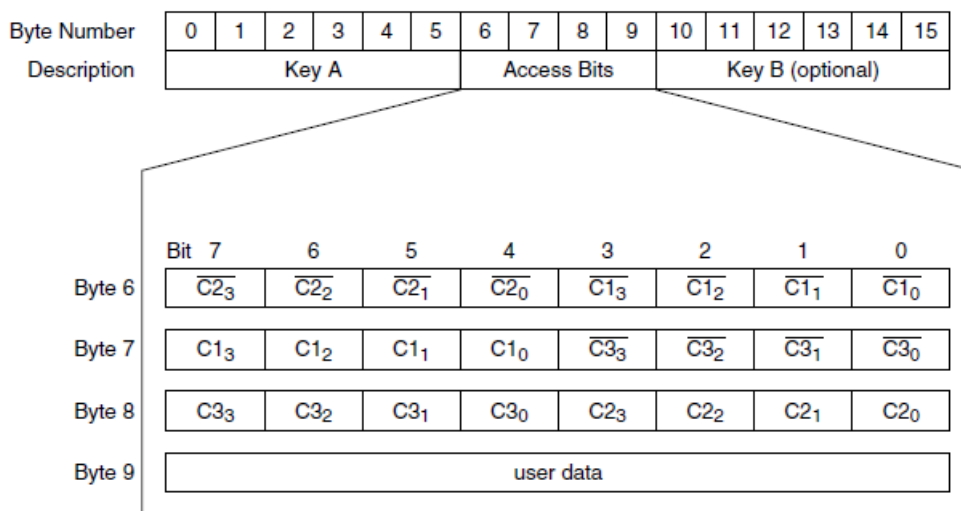
1. 讀、寫區塊：一般的資料保存，可以進行讀、寫操作
2. 數值區塊(value block)：用作資料值(4 Bytes)，可以進行加值、減值、讀值操作，例如電子錢包應用

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	value				value				value				adr	adr	adr	adr

3. 控制區塊：每個扇區的區塊 3 為控制區塊，包括密碼 A (6 位元組)、存取控制 (4 位元組)、密碼 B (6 位元組)，如下圖：

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Key A						Access Bits				Key B (optional)					

- 每個扇區(Sector)的密碼和存取控制都是獨立的，可以根據實際需要設定各自的密碼及存取控制。存取控制為 4 個位元組，共 32 位元，扇區中的每個區塊（包括資料區塊和控制區塊）的存取條件是由密碼和存取控制共同決定的，在存取控制中每個區塊都有對應的三個控制位元，其結構與對應之操作權限如下圖示：



Access Bits	Valid Commands		Block	Description
$C1_3 C2_3 C3_3$	read, write	→	3	sector trailer
$C1_2 C2_2 C3_2$	read, write, increment, decrement, transfer, restore	→	2	data block
$C1_1 C2_1 C3_1$	read, write, increment, decrement, transfer, restore	→	1	data block
$C1_0 C2_0 C3_0$	read, write, increment, decrement, transfer, restore	→	0	data block

而各區塊對應的存取位元定義如下：

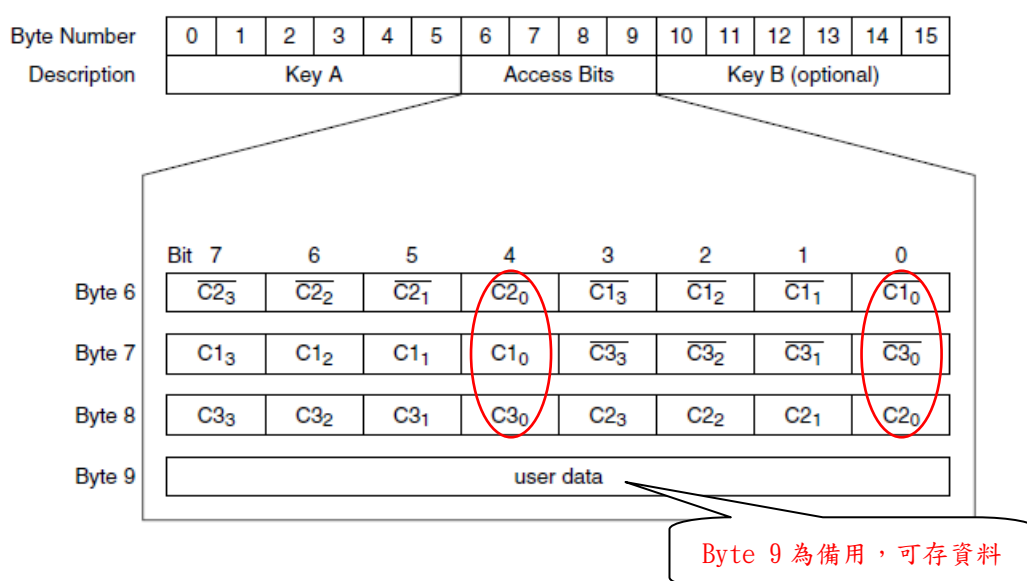
區塊 0：C10 C20 C30

區塊 1：C11 C21 C31

區塊 2：C12 C22 C32

區塊 3：C13 C23 C33

三個控制位元以正和反兩種形式存在於存取控制位元組中，決定了該區塊的存取權限（如進行減值操作必須驗證 KEY A，進行加值操作必須驗證 KEY B，等等）。三個控制位元在存取控制位元組中的位置，以區塊 0 為例：



– 資料區塊（0、1、2）的存取控制如下：

Access bits			Access condition			Application	
C1	C2	C3	Read	Write	Increment	Dec/Trans/Rest	
0	0	0	key A B	key A B	key A B	key A B	transport
0	1	0	key A B	never	never	never	read/write block
1	0	0	key A B	key B	never	never	read/write block
1	1	0	key A B	key B	key B	key A B	value block
0	0	1	key A B	never	never	key A B	value block
0	1	1	key B	key B	never	never	read/write block
1	0	1	key B	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

（Key A|B 表示密碼 A 或密碼 B，Never 表示任何條件下不能實現）

從表中可以看出：

C1C2C3=000(出廠預設值)時最寬鬆，驗證密碼 A 或密碼 B 後可以進行任何操作

C1C2C3=111 無論驗證哪個密碼都不能進行任何操作，相當於把對應的區塊凍結了

C1C2C3=010 和 C1C2C3=101 都是唯讀，如果對應的資料區塊寫入的是只能讀取(Read only)的訊息，可設為這兩種模式

C1C2C3=001 時只能讀和減值，電子錢包一般設為這種模式，比如公車與捷運的悠遊卡或一卡通，使用者只能查詢或扣錢，不能私下加值，加值的時候透過儲值設備，先改變控制位元使卡片可以充值，充完值會再改回來。

– 控制區塊 3 的存取控制與資料區塊(0、1、2) 不同，它的存取控制如下：

Access bits			Access condition for						Remarks
			KEY A		Access bits		Key B		
C1	C2	C3	Read	Write	Read	Write	Read	Write	
0	0	0	never	key A	key A	never	key A	key A	Key A is able to read key B
0	1	0	never	never	key A	never	key A	never	Key A is able to read key B
1	0	0	never	key B	key A B	never	never	key B	
1	1	0	never	never	key A B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key A is able to read key B
0	1	1	never	key B	key A B	key B	never	key B	
1	0	1	never	never	key A B	key B	never	never	
1	1	1	never	never	key A B	never	never	never	

從上表中可以看出：

密碼 A 是永遠也讀不出來的，如果用戶的資料區塊指定了驗證密碼 A 卻忘了密碼 A，也就意味著這個資料區塊作廢了，但本扇區其他資料區塊和其他扇區的資料區塊不受影響

存取控制的寫控制在設置時一定要小心，一旦弄成了“Never”，則整個扇區的存取條件再也無法改變

C1C2C3=001(出廠預設值)時最寬鬆，除了密碼 A 不能讀之外，驗證了密碼 A 其他讀寫操作都可以進行

當 C1C2C3=000、C1C2C3=010 和 C1C2C3=001 時，所有的操作都不使用密碼 B，這時候密碼 B 佔據的 6 個位元組可以提供給使用者作為普通資料存儲用

由於出廠時資料區塊控制位元的預設值是 C1C2C3=000，控制區塊的預設值是 C1C2C3=001，而 Byte9 一般是 69H，所以出廠白卡的控制位元組通常是 FF078069H.

讀取 RFID Tag 的 ID 範例程式(cardRead.ino)

```
#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

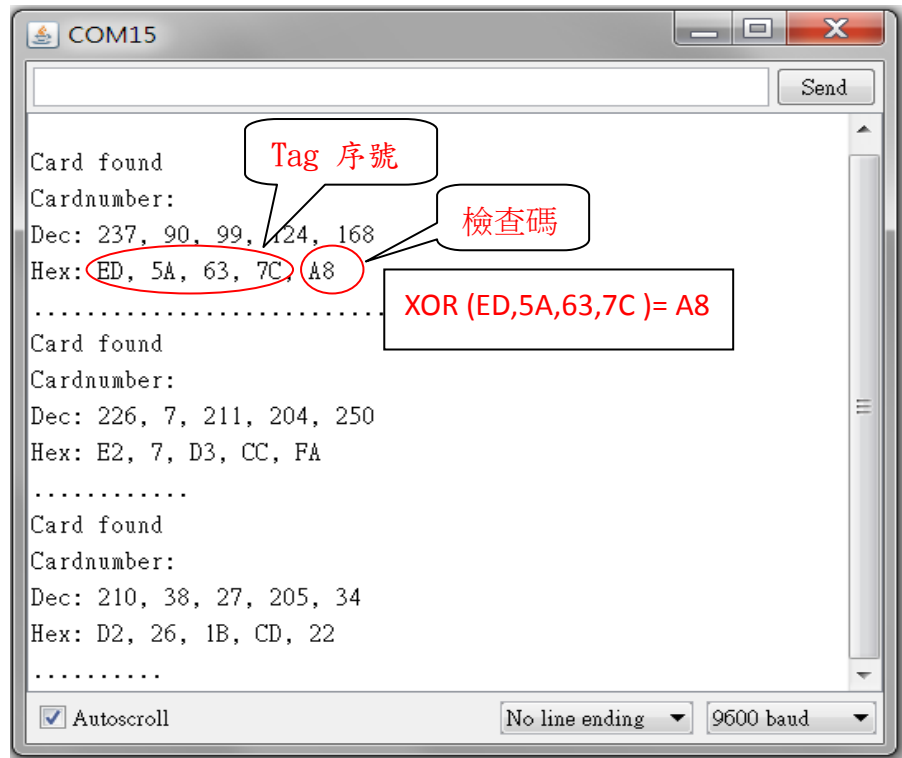
RFID rfid(SS_PIN, RST_PIN);

// Setup variables:
int serNum0;
int serNum1;
int serNum2;
int serNum3;
int serNum4;

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
}

void loop()
{
```

```
  if (rfid.isCard()) {
    if (rfid.readCardSerial()) {
      if (rfid.serNum[0] != serNum0
          && rfid.serNum[1] != serNum1
          && rfid.serNum[2] != serNum2
          && rfid.serNum[3] != serNum3
          && rfid.serNum[4] != serNum4
      ) {
        /* With a new cardnumber, show it. */
        Serial.println(" ");
        Serial.println("Card found");
        serNum0 = rfid.serNum[0];
        serNum1 = rfid.serNum[1];
        serNum2 = rfid.serNum[2];
```



```
serNum3 = rfid.serNum[3];
serNum4 = rfid.serNum[4];

//Serial.println(" ");
Serial.println("Cardnumber:");
Serial.print("Dec: ");
Serial.print(rfid.serNum[0], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[1], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[2], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[3], DEC);
Serial.print(", ");
Serial.print(rfid.serNum[4], DEC);
Serial.println(" ");

Serial.print("Hex: ");
Serial.print(rfid.serNum[0], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[1], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[2], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[3], HEX);
Serial.print(", ");
Serial.print(rfid.serNum[4], HEX);
Serial.println(" ");
} else {
    /* If we have the same ID, just write a dot. */
    Serial.print(".");
}
}

rfid.halt();
delay(200) ; // delay 200 ms
}
```

◆ RFID 讀寫 Block 範例程式(模擬儲值、消費、充值的情境)

◇ 主要流程如下：

1. 尋找卡片，回傳**卡片型態**代號
2. 呼叫防衝撞程序(Anti-collision loop in readCardSerial())，**回傳卡片序號**
3. 選擇作用卡片

寫入新密碼到區塊 11(控制區塊)，**寫入數值 100** 到區塊 8(資料區塊)的位址

4. **讀取**區塊 8 的資料
5. 讀取區塊 8，減 18，運算完成將結果重新寫入
6. 讀取區塊 8，加 10，運算完成將結果重新寫入
7. **中止**卡片存取，進入睡眠模式

```
/*
 * 文 件 名：RFID_blockRW_demo.ino
 * 功能描述：Mifare1 尋卡→防衝突→選卡→讀寫介面
 */

/*
 * Documentation：RFID_blockRW_demo.ino
 * Created By：Evan    WWW.B2CQSHOP.COM
 * Creation Date：2011.09.19
 * Modified：
 * Translated By：Gareth Halfacree    gareth.halfacree.co.uk
 * Description：Mifare, Find Cards → Anti-Collision → Read and Write Cards
 * Translated By：Tony Su/2015/04/16    www.appsduino.com
 * Description：simplified with rfid library
 */
// the sensor communicates using SPI, so include the library:
#include <SPI.h>
#include <RFID.h>

#define SS_PIN 10
#define RST_PIN 9

RFID rfid(SS_PIN, RST_PIN);

// Maximum length of an array
```

```
#define MAX_LEN 16

unsigned char  writeData[16]={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100}; //
Initialise $100
unsigned char  moneyConsume = 18 ; // Spend $18
unsigned char  moneyAdd = 10 ; // Recharge $10

// Sector A Password, 16 sectors, each sector password is 6-bytes
unsigned char sectorKeyA[16][16] = //舊密碼
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}, //當改變密碼後,需使用新密碼
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
};
unsigned char sectorNewKeyA[16][16] = //新密碼
{{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x84, 0x07, 0x15, 0x76, 0x14},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x33, 0x07, 0x15, 0x34, 0x14},
 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xff, 0x07, 0x80, 0x69, 0x19, 0x33, 0x07, 0x15, 0x34, 0x14},
};

void setup() {
    Serial.begin(9600);
    // start the SPI library:
    SPI.begin();
    rfid.init();
    delay(1000) ;
    Serial.println("RFID write/read block Example");
    Serial.println("Demo for Read/Write data in (Sector 2,Block 0) or block #8");
}

void loop()
{
    unsigned char i, tmp;
    unsigned char status;
    unsigned char str[MAX_LEN];
    unsigned char RC_size;
    unsigned char blockAddr; // Select the operating block address, 0-63
```



```
//-----  
    if (rfid.isCard()) { // 檢查是否偵測到 RFID Tag ? 並回傳 true/false  
        if (rfid.readCardSerial()) { //讀取 RFID 的 ID/序號(4 bytes)與檢查碼(1 byte)  
            /* With a new cardnumber, show it. */  
            Serial.println(" ");  
            Serial.println("Card found");  
            Serial.print("Serial #(Hex) : ");  
            Serial.print(rfid.serNum[0], HEX);  
            Serial.print(", ");  
            Serial.print(rfid.serNum[1], HEX);  
            Serial.print(", ");  
            Serial.print(rfid.serNum[2], HEX);  
            Serial.print(", ");  
            Serial.print(rfid.serNum[3], HEX);  
            Serial.print(", ");  
            Serial.print(rfid.serNum[4], HEX);  
            Serial.println(" ");  
        }  
    }  
//-----  
    // Election card, return capacity  
    RC_size = rfid.SelectTag(rfid.serNum); // 選擇一張卡並回傳卡片的容量  
    // Now a card is selected. The UID and SAK is in mfrc522.uid.  
    if (RC_size != 0) {  
        Serial.print("The size of the card is : ");  
        Serial.print(RC_size, DEC);  
        Serial.println("K bits");//單位 : K bits  
    }  
    // 存入$100 in Registration card  
    blockAddr = 11; //驗證 Data block 11 中的密碼 A  
    status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorKeyA[blockAddr/4],  
rfid.serNum);// Authentication by verify Key A  
    if (status == MI_OK)  
    {  
        // Write data  
        status = rfid.write(blockAddr, sectorNewKeyA[blockAddr/4]);//寫入新的密碼  
        Serial.print("set the new card password, and can modify the data of the Sector ");  
        Serial.print(blockAddr/4, DEC);  
        Serial.println(" : ");  
        for (i=0; i<6; i++)
```

```
{
    Serial.print(sectorNewKeyA[blockAddr/4][i], HEX);
    Serial.print(" , ");
}
Serial.println(" ");
blockAddr = blockAddr - 3 ;
status = rfid.write(blockAddr, writeData); //在 block #8 的 byte #15 存入$100
if(status == MI_OK)
{
    Serial.println("You are CyperShop VIP Member, The card has  $100 !");
}
}

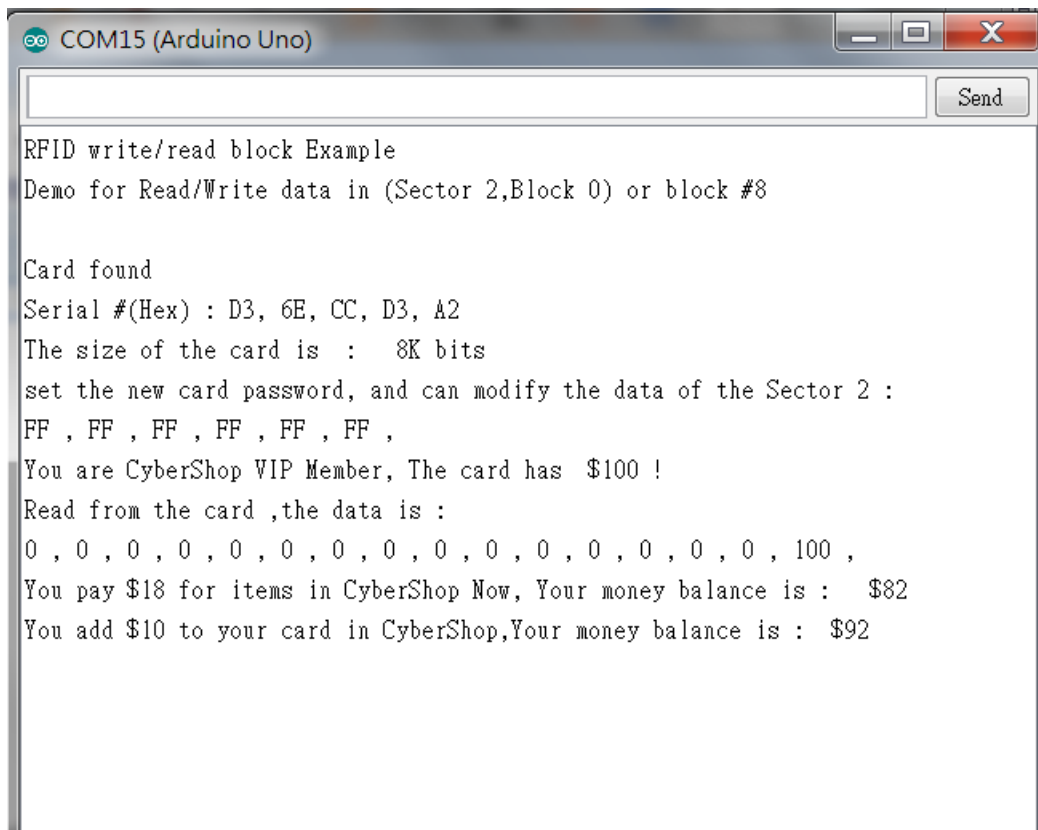
// Card reader
blockAddr = 11;    //驗證 Data block 11 中的密碼 A
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
rfid.serNum); // Authentication
if (status == MI_OK)
{
    // Read data
    blockAddr = blockAddr - 3 ;
    status = rfid.read(blockAddr, str); //讀取 Sector 2 的控制區塊資料(block #11)
    if (status == MI_OK)
    {
        Serial.println("Read from the card , the data is : ");
        for (i=0; i<16; i++)
        {
            Serial.print(str[i], DEC);
            Serial.print(" , ");
        }
        Serial.println(" ");
    }
}

// 模擬消費/Consumer
blockAddr = 11;    //驗證 Data block 11 中的密碼 A
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
rfid.serNum); // Authentication
if (status == MI_OK)
{
    // Read data
```

```
blockAddr = blockAddr - 3 ;
status = rfid.read(blockAddr, str);
if (status == MI_OK)
{
    if( str[15] < moneyConsume )
    {
        Serial.println(" The money is not enough !");
    }
    else
    {
        str[15] = str[15] - moneyConsume;
        status = rfid.write(blockAddr, str); //在 block #8 的 byte #15 減去$18 後再存入
        if(status == MI_OK)
        {
            Serial.print("You pay $18 for items in CyberShop Now, Your money balance
is :  $");
            Serial.print(str[15], DEC);
            Serial.println(" ");
        }
    }
}

// Recharge
blockAddr = 11; // Data block 11
status = rfid.auth(PICC_AUTHENT1A, blockAddr, sectorNewKeyA[blockAddr/4],
rfid.serNum); // Authentication
if (status == MI_OK)
{
    // Read data
    blockAddr = blockAddr - 3 ;
    status = rfid.read(blockAddr, str);
    if (status == MI_OK)
    {
        tmp = (int)(str[15] + moneyAdd) ;
        //Serial.println(tmp, DEC);
        if( tmp > 254 ) //一個 Byte 數值不能超過 255
        {
            Serial.println(" The money of the card can not be more than 255 !");
        }
    }
    else
```

```
{
  str[15] = str[15] + moneyAdd ;
  status = rfid.write(blockAddr, str);
  if(status == MI_OK)
  {
    Serial.print("You add $10 to your card in CyberShop, Your money balance is :  $");
    Serial.print(str[15], DEC);
    Serial.println(" ");
  }
}
}
}
rfid.halt(); // Enter Sleep Mode
delay(500) ; // waiting for 0.5 sec
}
```



```
COM15 (Arduino Uno)
RFID write/read block Example
Demo for Read/Write data in (Sector 2,Block 0) or block #8

Card found
Serial #(Hex) : D3, 6E, CC, D3, A2
The size of the card is : 8K bits
set the new card password, and can modify the data of the Sector 2 :
FF , FF , FF , FF , FF , FF ,
You are CyberShop VIP Member, The card has $100 !
Read from the card ,the data is :
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 100 ,
You pay $18 for items in CyberShop Now, Your money balance is : $82
You add $10 to your card in CyberShop,Your money balance is : $92
```

範例程式執行結果